SÍNTESIS AUTOMÁTICA DE SISTEMAS DIFUSOS MEDIANTE XFUZZY

Santiago Sánchez Solano[1], **María Brox Jiménez**[2]

[1] *Instituto de Microelectrónica de Sevilla, IMSE-CNM-CSIC*, santiago@imse-cnm.csic.es
[2] *Dpto. Arquitectura de Computadores, Universidad de Córdoba*, mbrox@uco.es

Resumen

En esta comunicación se describen las dos herramientas para síntesis hardware de sistemas de inferencia difusos incluidas en el entorno *Xfuzzy*. Ambas herramientas comparten una arquitectura de implementación común, aunque están basadas en metodologías de diseño diferentes. *Xfvhdl* (*Xfuzzy to VHDL*) genera código VHDL estándar que puede ser posteriormente implementado mediante un ASIC o una FPGA. *Xfsg (Xfuzzy to SysGen)* se apoya en el entorno Matlab/Simulink y las herramientas de desarrollo de sistemas de procesado digital de señal sobre FPGAs de Xilinx. Como demuestran los ejemplos considerados, el empleo de estas herramientas permite acelerar considerablemente las etapas de descripción, verificación y síntesis de sistemas de control basados en lógica difusa.

Palabras Clave: Sistemas de inferencia difusos, Herramientas de CAD, Implementación Hardware.

1 INTRODUCCIÓN

La capacidad de la lógica fuzzy para modelar el comportamiento de sistemas complejos, así como para describir estrategias de control basadas en reglas similares a las empleadas en el lenguaje natural, ha motivado su empleo en numerosas aplicaciones relacionadas con la automatización industrial, la robótica y otros campos de la ingeniería [7]. Las primeras realizaciones de sistemas de inferencia se realizaron mediante software ejecutado en procesadores de propósito general. Posteriormente, en la década de los 90, se desarrollaron una serie de 'coprocesadores difusos' cuyo objetivo era acelerar las operaciones necesarias para llevar a cabo el proceso de inferencia. Sin embargo, como consecuencia de los continuos avances en las tecnologías de fabricación de circuitos integrados, en los últimos años se han propuesto numerosas alternativas para la implementación hardware de sistemas difusos mediante ASIC o FPGAs [3], [4].

El importante éxito de la lógica fuzzy en las últimas décadas del siglo pasado motivó el desarrollo de numerosas herramientas dedicadas al diseño de este tipo de sistemas. En los primeros años, la mayoría del sofware disponible comercialmente estaba dirigido hacia la generación de código optimizado para las diferentes familias de microcontroladores de la época. Posteriormente aparecieron numerosas propuestas para la síntesis automática de sistemas difusos mediante hardware específico basado en técnicas de diseño analógicas y digitales [5], [6], [8]. Más recientemente, han surgido diferentes alternativas que aprovechan las herramientas incluidas en el entorno Matlab/Simulink para facilitar el diseño e implementación de sistemas difusos [1], [2], [9].

En este trabajo se describen las últimas versiones de las dos herramientas de síntesis hardware de sistemas difusos integradas en el entorno de desarrollo *Xfuzzy* [14]. Una de ellas, denominada *Xfvhdl*, constituye una actualización de la herramienta incluida en versiones anteriores del entorno, y traslada la especificación del sistema difuso en una descripción VHDL que puede ser implementada en las tecnologías de destino disponibles en las herramientas estándar de síntesis para ASICs o FPGAs. La segunda herramienta, *Xfsg*, que utiliza el software SysGen para desarrollo sobre FPGAs de Xilinx, permite completar el flujo de diseño sobre el entorno Matlab/Simulink y aprovecha todas las opciones de configurabilidad y flexibilidad que ofrece dicho entorno.

En la Sección 2 se introducen las principales características del entorno *Xfuzzy* y sus diferentes herramientas. La estructura de circuitos en la que se basan las dos herramientas de síntesis es descrita en la Sección 3. Las funcionalidades y opciones de ambas herramientas se detallan en las Secciones 4 y 5. La Sección 6 ilustra un ejemplo de aplicación a un problema clásico de control. Por último, las principales conclusiones obtenidas tras la realización del trabajo son resumidas en la Sección 7.

2 EL ENTORNO *XFUZZY*

El entorno de desarrollo de sistemas difusos *Xfuzzy* proporciona un conjunto de herramientas que facilitan las sucesivas fases de descripción, verificación y síntesis de sistemas de inferencia basados en lógica difusa. Un sistema difuso se describe en *Xfuzzy* mediante una especificación XFL3 que combina bases de reglas fuzzy y módulos crisp (para realizar tareas de inferencia e implementar bloques lógicos y aritméticos, respectivamente). Las bases de conocimiento pueden ser definidas directamente con ayuda de la interfaz gráfica de usuario (*Xfedit*), o extraídas a partir de datos numéricos usando herramientas de minería de datos (*Xfdm*). Tanto las funciones de pertenencia como las bases de reglas pueden ser simplificadas para mejorar su interpretabilidad lingüística (*Xfsp*). También es posible ajustar los parámetros del sistema usando diferentes algoritmos de aprendizaje supervisado (*Xfsl*). La verificación funcional del sistema puede llevarse a cabo mediante dos herramientas del entorno. La primera de ellas permite analizar la relación entrada/salida del sistema (*Xfplot*), mientras que la segunda permite simular su comportamiento en lazo cerrado en combinación con un modelo Java de la planta (*Xfsim*). Una vez validada, la especificación puede trasladarse a código C, C++ o Java para obtener una implementación software del sistema de inferencia.

La Figura 1 ilustra la interfaz gráfica de usuario de *Xfuzzy* cuando se utiliza para definir un controlador difuso para el problema del doble integrador. El sistema está formado en este caso por dos bases de reglas y un bloque crisp que realiza la operación aritmética de sustracción.

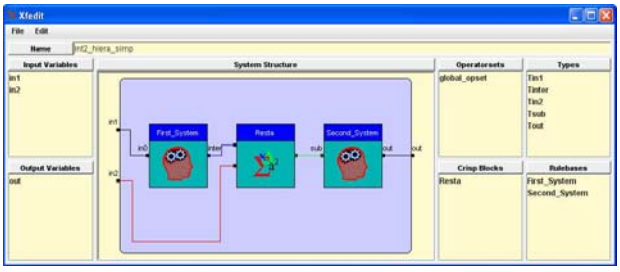


Figura 1. Descripción en *Xfuzzy* de un controlador difuso para el problema del doble integrador.

3 ARQUITECTURA BASADA EN REGLAS ACTIVAS

Las herramientas de síntesis hardware incluidas en *Xfuzzy* facilitan la implementación microelectrónica de sistemas difusos de acuerdo con la arquitectura mostrada en la Figura 2. Las características que definen esta arquitectura son la limitación del grado de solapamiento de las funciones de pertenencia de los antecedentes, la utilización de

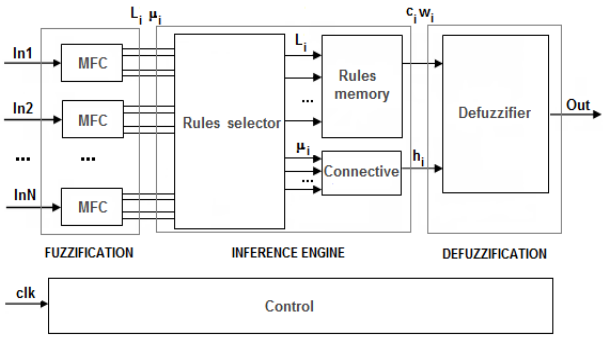


Figura 2. Arquitectura de reglas activas para la implementación hardware de sistemas difusos.

métodos de defuzzificación simplificados y el empleo de un esquema de procesado basado en reglas activas [11]. En la Figura 2 aparecen las tres etapas típicas en el cálculo de una inferencia difusa: fuzzificación, inferencia y defuzzificación. En la etapa de fuzzificación los circuitos generadores de funciones de pertenencia (MFC) suministran las parejas de datos etiqueta-grado de pertenencia (L_i, μ_i) para cada valor de las entradas del sistema. Las distintas combinaciones de estas etiquetas determinarán las reglas activas. En la etapa de inferencia se procesan secuencialmente cada una de las reglas activas, utilizándose para ello un circuito selector de reglas activas formado por un conjunto de multiplexores controlados por un contador. En cada ciclo de reloj los grados de pertenencia μi de cada una de las entradas son combinados a través del conectivo de antecedentes para calcular el grado de activación de la regla (α'), mientras que las respectivas etiquetas de los antecedentes direccionan la posición de memoria que contiene los parámetros (c_r, w_r) que representan su correspondiente consecuente. En la etapa de defuzzificación se calcula el valor de salida procesando los consecuentes de cada regla con sus diferentes grados de activación de acuerdo con el método de defuzzificación elegido. Finalmente un bloque de control se encarga de regular la temporización del proceso de inferencia.

Esta arquitectura se caracteriza por ser altamente configurable debido a la disponibilidad de diferentes opciones para implementar cada uno de los bloques que la forman. Los MFCs pueden ser implementados mediante almacenamiento en memoria o cálculo aritmético. La estrategia basada en almacenamiento en memoria almacena en cada localización los valores de las etiquetas y los grados de pertenencia correspondientes a cada punto del universo de discurso. Por otro lado, la estrategia basada en técnicas aritméticas emplea circuitos aritméticos para implementar familias de funciones triangulares normalizadas (correspondientes a los tipos *triangular* y *sh_triangular* definidos en XFL3) usando una memoria que almacena los puntos de corte y pendientes para cada antecedente. El diseñador puede elegir también implementar los conectivos de antecedentes mediante operadores mínimo o pro-

ducto. Finalmente, en la etapa de defuzzificación, dependiendo de si las bases de reglas corresponden a aproximadores o elementos de toma de decisiones, se pueden emplear diferentes defuzzificadores: FuzzyMean, Weighted-FuzzyMean, Takagi-Sugeno de orden uno o MaxLabel.

4 HERRAMIENTA *XFVHDL*

Para automatizar el proceso de diseño con la arquitectura descrita en la sección anterior, la técnica basada en VHDL utiliza una librería de bloques funcionales parametrizables y sintetizables descritos mediante lenguaje VHDL (la librería *XfuzzyLib_src*). La librería incluye bloques correspondientes a los distintos elementos de la arquitectura mostrada en la Figura 2, además de un conjunto de bloques crisps que implementan operaciones aritméticas y funciones lógicas. La construcción de un sistema difuso mediante esta técnica requiere seleccionar e interconectar los bloques necesarios, así como generar las descripciones VHDL que definen los parámetros de las bases de conocimiento. El uso del lenguaje VHDL, y el hecho de que las descripciones cumplen con las restricciones impuestas por las principales herramientas de síntesis de circuitos integrados, permite que con esta librería puedan diseñarse tanto ASICs como FPGAs siguiendo el flujo de desarrollo habitual en este tipo de herramientas.

La herramienta de síntesis *Xfvhdl* facilita la traslación automática de una especificación XFL3 en una descripción VHDL acorde con esta técnica de diseño y que puede ser implementada directamente o incluida como componente de un sistema empotrado. La interfaz gráfica de usuario de esta herramienta se muestra en la Figura 3. La parte superior de la ventana recoge información sobre los ficheros y directorios utilizados. La zona central izquierda muestra la base de conocimiento estructurada en componentes que son agrupados bajo las categorías “RuleBases” y “CrispBlocks”. Cuando se selecciona una base de reglas, la zona de la derecha de la interfaz muestra información extraída de la especificación XFL3 sobre las funciones de pertenencia y las reglas. En esta misma zona, el usuario puede definir los parámetros relacionados con la dimensión del sistema, así como seleccionar la estrategia de implementación de los antecedentes.

Las herramientas de diseño de FPGA permiten normalmente inferir el uso de memoria a partir de descripciones VHDL genéricas en la etapa de síntesis y seleccionar el tipo de memoria a utilizar en la etapa de implementación. Esta característica permite considerar distintas opciones de implementación para los antecedentes y la base de reglas de un sistema difuso. Por este motivo, en la parte derecha de la interfaz también puede seleccionarse el tipo de memoria (RAM, ROM o bloque lógico) utilizada en ambos casos y la técnica empleada en los MFCs.

Cuando las opciones arquitecturales y los parámetros relacionados con el tamaño de los buses han sido definidos para todos los componentes del sistema, aparece una marca verde en el nivel superior de la jerarquía y se activan los diferentes botones de la zona inferior de la ventana. En particular, es posible generar los ficheros de salida presionando el botón “*Generate VHDL code*”. Básicamente se genera una descripción VHDL y un testbench

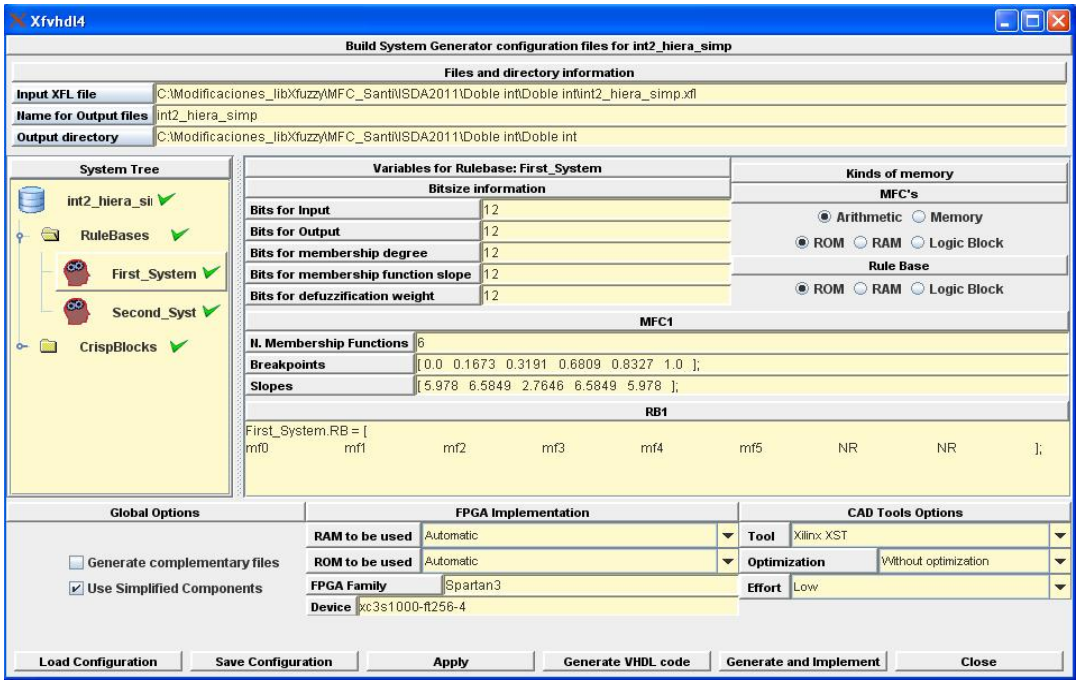


Figura 3. Interfaz gráfica de la herramienta *Xfvhdl*.

(descrito también en VHDL para verificar la funcionalidad del módulo) para cada una de las bases de reglas, junto con la descripción VHDL y un testbench para el nivel superior de la jerarquía.

Para cada una de las bases de reglas, los bloques correspondientes al método de defuzzificación o el operador usado como conectivo se extraen directamente de la especificación XFL3. Sin embargo, el estilo de la descripción VHDL y los componentes que incluye dependen de las opciones arquitecturales escogidas. Concretamente, si se selecciona la opción de bloque lógico para las memorias de antecedentes o las de reglas, la descripción generada utiliza una estructura tipo CASE. Cuando la opción escogida es ROM, se genera una estructura tal que, si el usuario selecciona la opción correspondiente, la herramienta de síntesis puede extraer la memoria ROM para implementar estas descripciones. Finalmente, si la opción seleccionada es RAM, se incluye un bloque de la librería *XfuzzyLib_src* que será identificado como una memoria RAM por la mayoría de herrarmienta de síntesis.

Para el caso de implementación en FPGAs de Xilinx, es posible seleccionar el tipo de memoria (BRAM o CLBs) que va a ser utilizada en el dispositivo programable. Esta elección, junto con otras opciones relativas al dispositivo y herramientas de síntesis empleadas, puede llevarse a cabo mediante los campos que aparecen en la zona inferior de la ventana principal de *Xfvhdl*. Además de las descripciones VHDL, la herramienta genera una serie de ficheros que facilitan la síntesis e implementación del sistema sobre FPGAs, y permite lanzar ambos procesos desde la interfaz de usuario mediante el botón "*Generate and implement*".

5 HERRAMIENTA *XFSG*

La segunda técnica de síntesis considerada está basada en SysGen, la herramienta para desarrollo de sistemas de procesado digital de señal sobre FPGAs de Xilinx. Esta herramienta está integrada en el entorno Matlab e incluye una librería de módulos de Simulink, denominada "Xilinx Blockset", así como el software necesario para trasladar un modelo Simulink en una descripción de hardware que puede ser implementada en una FPGA [13].

Haciendo uso de esta librería y utilizando las facilidades proporcionadas por SysGen, se ha generado una librería de módulos especializados, *XfuzzyLib*, compuesta por bloques que cubren las tres etapas de fuzzificación, inferencia y defuzzificación de la arquitectura de sistemas difusos descrita en la Sección 3. A partir de los bloques de esta librería, la construcción de un sistema difuso requiere la elección, interconexión y parametrización de los bloques necesarios. Los distintos módulos de la librería "Xilinx Blockset" admiten una serie de parámetros que definen aspectos tales como su funcionalidad, tamaño o tipo de aritmética utilizada. Del mismo modo, una vez realizado el diagrama de bloques de un nuevo módulo de la librería *XfuzzyLib*, se encapsula como un "subsistema" y se añade una "máscara" para identificar los distintos parámetros que caracterizan al módulo. De esta forma, cuando dicho subsistema es usado en un nivel jerárquico superior, estos parámetros pueden asignarse mediante valores numéricos o mediante variables a las que se les asigna un valor numérico en la ventana de comandos de Matlab o a través de un fichero ".m". La funcionalidad del diseño puede verificarse en todo momento con ayuda del simulador Simulink y sus diferentes facilidades para generar señales de excitación y para capturar y representar gráficamente los datos de salida.

Además de los bloques que constituyen los elementos constructivos básicos de un sistema difuso, la librería *XfuzzyLib* incluye bloques con descripciones de arquitecturas tipo de módulos de inferencia difusos (FLCs) que se diferencian entre sí en el número de entradas, el conectivo que usan para calcular el grado de activación de las reglas y el método de defuzzificación empleado. Es posible combinar de forma jerárquica estos módulos de inferencia simples para definir sistemas difusos de mayor complejidad. Los bloques que describen arquitecturas de FLCs son totalmente parametrizables, por lo que el diseñador de sistemas difusos puede adaptar la funcionalidad del módulo a los requerimientos del problema planteado sin más que definir los parámetros adecuados. Básicamente hay dos tipos de parámetros: los que corresponden al dimensionado del sistema de inferencia, como son el número de bits de las entradas, salidas y grados de pertenencia, y los que definen la base de conocimiento del mismo, es decir, las funciones de pertenencia y la base de reglas. Para facilitar su uso por parte del diseñador, los distintos parámetros corresponden a variables y estructuras de datos a las que pueden asignarse valores a través de la propia interfaz gráfica de Simulink o utilizando un fichero ".m" de Matlab. También a este nivel es posible aprovechar toda la funcionalidad del entorno Matlab para verificar la funcionalidad del sistema de inferencia [12].

La herramienta *Xfsg*, incorporada al entorno de desarrollo *Xfuzzy*, permite generar los ficheros requeridos para automatizar el flujo de diseño descrito anteriormente. La interfaz gráfica de usuario de esta herramienta es muy similar a la de *Xfvhdl*. Una vez identificados todos los componentes del sistema difuso, *Xfsg* genera un fichero ".mdl", que contiene el modelo Simulink del controlador difuso, y un fichero ".m", que contiene los parámetros que definen el tamaño y la funcionalidad de los diferentes componentes del sistema de inferencia. El modelo obtenido incluye un bloque "System Generator" que facilita la síntesis hardware del diseño descrito en Simulink mediante la traslación del modelo a diferentes tipos de netlists. También puede generar el fichero de programación de la FPGA, e incluso permite la verificación funcional del controlador en lazo cerrado a través de la cosimulación hardware/software de la implementación del sistema de control junto a un modelo de la planta.

6 APLICACIÓN A UN PROBLEMA DE CONTROL

Las herramientas de síntesis descritas anteriormente han sido utilizadas para la implementación de sistemas difusos aplicados a un problema típico dentro de la ingeniería de control, el doble integrador. La metodología de diseño descrita en esta sección combina el uso de las herramientas de síntesis hardware del entorno *Xfuzzy*, herramientas de síntesis e implementación de FPGAs de Xilinx, y herramientas de simulación y modelado del entorno Matlab/ Simulink. El flujo de diseño se ilustra en la Figura 4.

La descripción del controlador en *Xfuzzy*, correspondiente al sistema mostrado en la Figura 1, constituye la entrada a las dos herramientas de síntesis. La funcionalidad de los resultados obtenidos con ambas herramientas pueden ser verificadas usando las facilidades gráficas y de simulación de ModelSim (Figura 5b) y Matlab (Figura 5c), respectivamente, y comparadas con la prevista en el entorno *Xfuzzy* (Figura 5a). Como se muestra en la Figura 4 las metodologías de ambas técnicas de diseño siguen a priori flujos diferentes. Sin embargo, Simulink permite desarrollar un modelo en el que el código VHDL generado por *Xfvhdl* puede ser incluido en un bloque "Black Box" junto con un bloque ModelSim, ambos proporcionados por la librería "Xilinx Blockset", para realizar una cosimulación HDL.

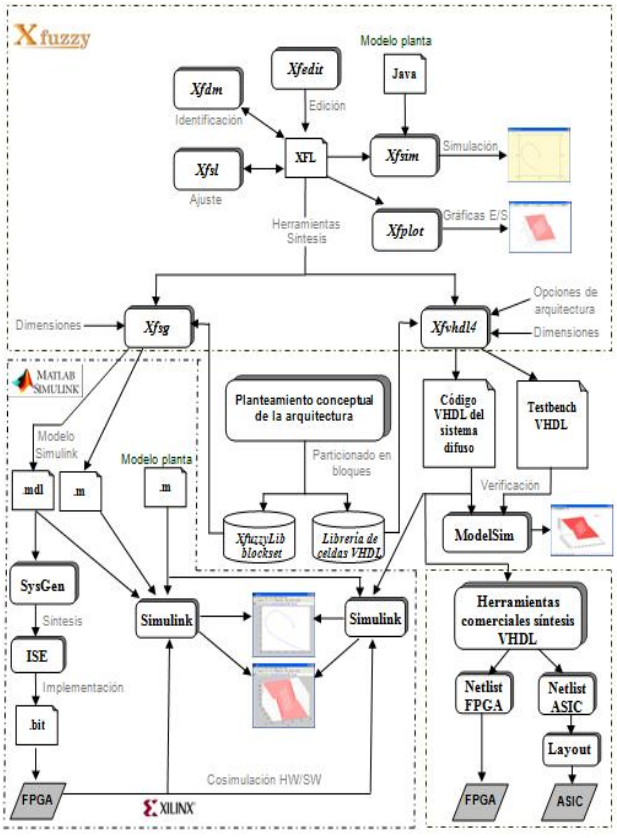


Figura 4. Flujo de diseño soportado por las dos herramientas de síntesis hardware de *Xfuzzy*.

Finalmente también es posible realizar en Simulink la verificación funcional en lazo cerrado de cualquiera de las dos implementaciones del controlador mediante la cosimulación de un modelo software de la planta descrito en Matlab y la implementación hardware del controlador en una FPGA.

La realización hardware de este controlador usando técnicas aritméticas en los antecedentes y memoria ROM de tipo distribuida (con 12 bits de precisión en las entradas y salidas para todas las bases de reglas) consume 185 Slices con *Xfvhdl* y 259 con *Xfsg* (aproximadamente el 3% y 4%, respectivamente, de los Slices disponibles en una FPGA Spartan 3A de Xilinx). El controlador también emplea para ambas técnicas 4 de los 20 multiplicadores disponibles en la FPGA.

7 CONCLUSIONES

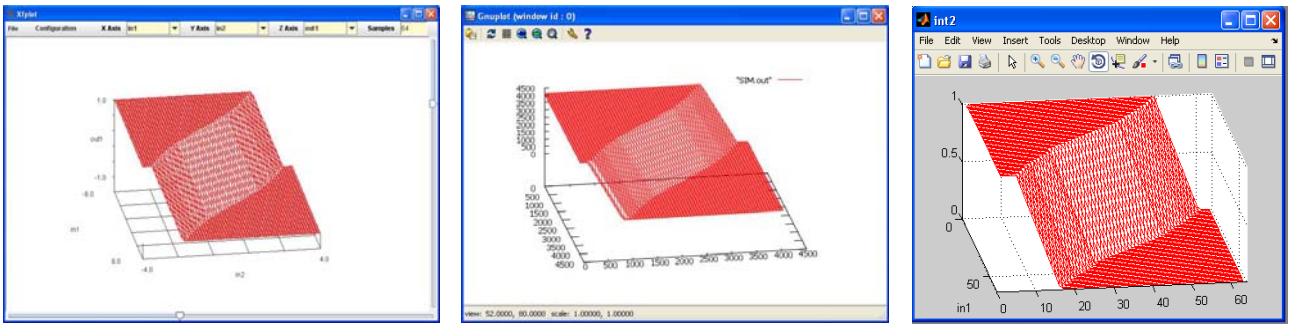
Las dos herramientas de síntesis automática de sistemas de inferencia fuzzy presentadas en esta comunicación permiten poner de manifiesto que la disponibilidad de flujos de diseño, soportados por el uso de librerías de celdas parametrizadas y herramientas de CAD, acelera considerablemente la implementación hardware de sistemas difusos, facilitando la exploración del espacio de diseño para una determinada aplicación. Una de las herramientas descritas está enfocada hacia la implementación hardware de sistemas difusos sobre FPGAs de Xilinx, mientras que la otra proporciona código VHDL sintetizable sobre cualquier tecnología (FPGA o ASIC) disponible para las herramientas de síntesis comerciales.

Comparadas con las versiones previas de las herramientas de síntesis de *Xfuzzy* [10], las nuevas herramientas proporcionan una funcionalidad mejorada de la mayoría de los componentes incluidos en las librerías VHDL y Simulink, la implementación de algunos de los tipos de familias de funciones de pertenencia definidos en el lenguaje de especificación XFL3, nuevos operadores y métodos de defuzzificación y, como avance más significativo, la implementación directa de sistemas difusos complejos compuestos por una combinación de bases de reglas y módulos crisp.

Las ventajas de las herramientas y técnicas de diseño propuestas han quedado demostradas mediante su aplicación en un ejemplo de diseño.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Comunidad Europea con el proyecto MOBY-DIC FP7-IST-248858, el Ministerio de Ciencia e Innovación con el proyecto TEC2008-04920, y la Junta de Andalucía con el proyecto P08-TIC-03674 (con soporte FEDER).



(a) (b) (c)

Figura 5. Superficies de control obtenidas por: a) *Xfuzzy*. b) *Xfvhdl*. c) *Xfsg*

Referencias

[1] I. H. Altas, A. M. Sharaf: A Generalized Direct Approach for Designing Fuzzy Logic Controllers in Matlab/Simulink GUI Environment. *International Journal of Information Technology and Intelligent Computing*, n.4 vol.1, 2007.

[2] A. Bakhti, L. Benbaouche: Simulink-DSP Co-Design of a Fuzzy Logic Controller. In *Proc. Industrial Electronics Society Annual Conference*, vol.1, pp. 4587-4592. Nov. 2006.

[3] K. Basterretxea, I. del Campo: Electronic hardware for fuzzy computation, in Laurent, A., and Lessot, M. J. (Eds.): *Scalable Fuzzy Algorithms for Data Management and Analysis: Methods and Design* (Information Science Reference, 2009), pp. 1-30.

[4] I. Baturone, A. Barriga, S. Sánchez-Solano, C. J. Jiménez, D. López: *Microelectronic Design of Fuzzy Logic-Based Systems*, CRC Press, 2000.

[5] R. G. Carvajal, A. Torralba, L.G. Franquelo: AFAN: a tool for the automatic synthesis of neural and fuzzy controllers with architecture optimization. In *Proc. International Symposium on Circuits and Systems*, vol. 1, pp. 637-640. 1997.

[6] T. Hollstein, S. K. Halgamuge, M. Glesner: Computer-Aided Design of Fuzzy Systems Based on Generic VHDL Specifications. *IEEE Transactions on Fuzzy Systems*, vol. 4, n. 4, pp. 403-417. 1996.

[7] J. Jarris: *Fuzzy logic applications in engineering science*. Springer Verlag, 2006.

[8] D. Kim, In-Hyun Cho: FADIS: An Integrated Development Environment for Automatic Design and Implementation of FLC. In *Proc. 1997 Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 33-39. Sep. 1997.

[9] O. Kobyrynka, Y. Stekh, O. Markelov: Comparison analysis of methods implemented in MATHLAB for fuzzy logic algorithms: In *Proc. 2011 CAD Systems and Microelectronics*, pp. 239-240. Feb. 2011.

[10] E. Lago, C. J. Jiménez, D. R. López, S. Sánchez-Solano, A. Barriga: XFVHDL: A Tool for the Synthesis of Fuzzy Logic Controllers, In *Proc. Design, Automation and Test in Europe*, pp. 102-107, Feb. 1998.

[11] S. Sánchez-Solano, A. Cabrera, I. Baturone, F. J. Moreno-Velo, M. Brox: Implementation of Embedded Fuzzy Controllers for Robotic Applications, *IEEE Trans. on Industrial Electronics*, vol. 54, n. 4, pp. 1937-1945. Aug 2007.

[12] S. Sánchez-Solano, E. del Toro, M. Brox, I. Baturone, A. Barriga: A Design Environment for Synthesis of Embedded Fuzzy Controllers on FPGAs" In *Proc. IEEE World Congress on Computational Intelligence*, pp. 44-51, Jul. 2010.

[13] System Generator for DSP User Guide, v10.1, Xilinx Inc., Available: http://www.xilinx.com

[14] *Xfuzzy*: Fuzzy Logic Design Tools, IMSE-CNM. Available: http://www.imse-cnm.csic.es/Xfuzzy